```
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRR   RRR        MMM         MMM              SSS
RRR   RRR        MMM         MMM              SSS
RRR   RRR        MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
```

```
**FILE**ID**RMSFILSTR                                   C  4



RRRRRRRR    MM      MM    SSSSSSSS  FFFFFFFFFF    IIIIII    LL            SSSSSSSS  TTTTTTTTTT  RRRRRRRR
RRRRRRRR    MM      MM    SSSSSSSS  FFFFFFFFFF    IIIIII    LL            SSSSSSSS  TTTTTTTTTT  RRRRRRRR
RR      RR  MMMM  MMMM  SS          FF              II      LL          SS              TT      RR      RR
RR      RR  MMMM  MMMM  SS          FF              II      LL          SS              TT      RR      RR
RR      RR  MM  MM  MM  SS          FF              II      LL          SS              TT      RR      RR
RR      RR  MM  MM  MM  SS          FF              II      LL          SS              TT      RR      RR
RRRRRRRR    MM      MM    SSSSSS    FFFFFFFF        II      LL            SSSSSS        TT      RRRRRRRR
RRRRRRRR    MM      MM    SSSSSS    FFFFFFFF        II      LL            SSSSSS        TT      RRRRRRRR
RR  RR      MM      MM          SS  FF              II      LL                  SS      TT      RR  RR
RR  RR      MM      MM          SS  FF              II      LL                  SS      TT      RR  RR
RR    RR    MM      MM          SS  FF              II      LL                  SS      TT      RR    RR        ....
RR    RR    MM      MM          SS  FF              II      LL                  SS      TT      RR    RR        ....
RR      RR  MM      MM  SSSSSSSS    FF            IIIIII    LLLLLLLLLL  SSSSSSSS        TT      RR      RR      ....
RR      RR  MM      MM  SSSSSSSS    FF            IIIIII    LLLLLLLLLL  SSSSSSSS        TT      RR      RR      ....


  SSSSSSSS  DDDDDDDD    LL
  SSSSSSSS  DDDDDDDD    LL
SS          DD      DD  LL
SS          DD      DD  LL
SS          DD      DD  LL
SS          DD      DD  LL
  SSSSSS    DD      DD  LL
  SSSSSS    DD      DD  LL
        SS  DD      DD  LL
        SS  DD      DD  LL
        SS  DD      DD  LL
        SS  DD      DD  LL
SSSSSSSS    DDDDDDDD    LLLLLLLLLL
SSSSSSSS    DDDDDDDD    LLLLLLLLLL
```

```
RMS

end
end_
```

```
{       $begin  rmsfilstr,V04-000
{
{****************************************************************************
{*                                                                          *
{*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
{*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
{*  ALL RIGHTS RESERVED.                                                    *
{*                                                                          *
{*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
{*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
{*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
{*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
{*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
{*  TRANSFERRED.                                                            *
{*                                                                          *
{*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
{*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
{*  CORPORATION.                                                            *
{*                                                                          *
{*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
{*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
{*                                                                          *
{*                                                                          *
{****************************************************************************
{
{       internal rms file structure definitions
{
{       Modified By:
{
{       V03-019 LJA0088         Laurie J. Anderson      17-Aug-1983
{               Add new field for AT journals - RJR$L_AT_CTX.  This
{               will allow users, if they choose to fill in this user field,
{               to have it written to the AT journal and thus printed out
{               by ANAL/AUDIT for them.
{
{       V03-018 KPL0008         Peter Lieberwirth       27-Jul-1983
{               Add fields to EXTEND RJR entry so RMS Recovery can use
{               RMS to re-do the extend.
{
{       V03-017 KPL0007         Peter Lieberwirth       22-Jun-1983
{               Correct misspelling in V03-016.
{
{       V03-016 KPL0006         Peter Lieberwirth       22-Jun-1983
{               Add mask longword to CREATE RJR entry - used to identify
{               which attributes are journaled in the entry.  REC_ATTR
{               field is really 32 bytes long, not 56.
{
{       V03-015 KPL0005         Peter Lieberwirth        7-Jun-1983
{               Longword align the RJR where it isn't.
{
{       V03-014 KPL0004         Peter Lieberwirth       26-May-1983
{               Fix bugs introduced in V03-013.
{
{       V03-013 KPL0003         Peter Lieberwirth       24-May-1983
{               Make RJB more robust by separating entry_type from
{               organization.  Make a filename entry common to AT,
```

```
RMSF

modu
/*
/*
/* c
/*
/*

aggr















end

end_
```

```
{               AI, BI, RU, mapping, and CREATE.  Add TPT for sequential
{               support.  Common EXTEND entry also (AT, AI).  Common
{               block IO entry Miscellaneous cleanup also.
{
{       V03-012 KPL0002         Peter Lieberwirth       28-Apr-1983
{               Add EXTEND journal entry description to RJR.
{
{       V03-011 KPL0001         Peter Lieberwirth       28-Apr-1983
{               Add AT journal entry descriptions to RJR.
{
{       V03-010 JWH0209         Jeffrey W. Horn         12-Apr-1983
{               Replace Mapping with JNLID.
{
{       V03-009 DAS0011         David Solomon           01-Apr-1983
{               Add RJR$W_JBKT_SIZE (size of journaled bucket).
{
{       V03-008 RAS0136         Ron Schaefer            17-Mar-1983
{               Add RJR$W_BKT_SIZE field to permit journaling of
{               partial buckets.
{
{       V03-007 RAS0131         Ron Schaefer            14-Mar-1983
{               Add RJR structure for accesibility to other RMS-related
{               facilities.  Combine RMSR structure with RJR.
{               Also change MCN0009 to a slightly smaller value.
{
{       V03-006 MCN0009         Maria del C. Nasr       07-Mar-1983
{               Define symbolic constant for maximum bucket size.
{
{       V03-005 TMK0003         Todd M. Katz            18-Dec-1982
{               Add the bit definitions for IRC$V_RU_DELETE and IRC$V_RU_UPDATE
{               to the IRC control byte. These bits are defined only for
{               prologue 3 files being recovery unit journalled.
{
{       V03-004 PCA1003         Paul C. Anagnostopoulos  4-Nov-1982
{               Add new quadword key information to the key descriptor.
{               Add new total area allocation field to area descriptor.
{
{       V03-003 MCN0008         Maria del C. Nasr       19-Oct-1982
{               KEY_COMPR flag in the key descriptor can be defined for
{               all keys.  Also eliminate COUNT_DUP, NORFA, and
{               PRG_D_RFA flags.
{
{       V03-002 TMK0002         Todd M. Katz            06-Sep-1982
{
{               1. Eliminate the following bits from the IRC definition:
{                  IRV$V_DEL_COUNT, IRC$V_KEYDELETE, IRC$V_HIGH_KEY,
{                  IRC$V_LAST_CONT, and IRC$V_FLATTEN.
{
{               2. Add the bit definition IRC$V_NODUPCNT within the IRC
{                  record control byte.
{
{               3. Redo all the IRC bits.
{
{               4. Add the constants IRC$C_DCNTSZFLD, IRC$C_SDROVHSZ3,
{                  IRC$C_RRVOVHDSZ, and IRC$C_RRVOVHSZ3.
{
```

```
RMS

modu
/*
/*
/*
/*
/*
/*
/*

agg
```

```
{               5. Add the bit IRC$V_FIRST_KEY to the record control byte
{                  definition.
{
{               6. Delete the bit definitions for BKT$V_LAST_CONT, and
{                  BKT$V_CONT_BKT from the index bucket definition.
{
{               7. Finally those bits reserved for RMS-11 use, but which are
{                  not used or referenced in RMS-32 code (bits number 5 and 6)
{                  have been marked as such, and their symbols (such as
{                  IRC$V_KEYDELETE) have been eliminated.
{
{       V03-001 TMK0001         Todd M. Katz            15-Jun-1982
{
{               1. Define the bit CONT_BKT within the bucket control byte.
{               2. Eliminate DUP_COUNT field from bucket definition and
{                  change the constant DUPBKTOVH from 8 to 4 to reflect this.
{               3. Define the bit FLATTEN within the record control byte.
{               4. Add the Prologue 3 SIDR overhead constants DUPSDROVH and
{                  SDROVHSZ3 to the index record definition.
{
{       V02-061 CDS0001         C Saether               23-Dec-1981
{               Add default global buffer count field to prologue
{               vbn 1 for relative and isam files.
{
{       V02-060 KBT0001         Keith B Thompson        12-Nov-1981
{               Add ONC and CBT flags in the AOP area descriptor field
{               and correct the position of CTG.
{
{       V02-059 PSK0004         Paulina S. Knibbe       23-Oct-1981
{               Remove LCB_EXITS flag.
{
{       V02-058 MCN0007         Maria del C. Nasr       15-May-1981
{               Add compression constants.
{
{       V02-057 PSK0003         Paulina S. Knibbe       16-Apr-1981
{               Add KEY_COMPR and REC_COMPR bits to the KEY structure
{               Add constants for largest non-compressed data record,
{               primary key and index (SIDR key.
{
{       V02-056 PSK0002         Paulina S. Knibbe       19-Mar-1981
{               Add pointer size field to the BKT structure
{
{       V02-055 MCN0006         Maria del C. Nasr       16-Mar-1981
{               Modify the BKT, and IRC structures to increase record id
{               size to a word, and reflect other changes required for the
{               new ISAM structure.
{
{       V02-054 PSK0001         Paulina Knibbe  11-Mar-1981
{               Add definitions for per-segment datatypes to the KEY structure
{               Add another prologue version constant
{
{       V02-053 ras0053         Ron Schaefer    24-Nov-1980     14:48
{               merge RMSINT and RMSIDXSTR file structure definitions into
{               one file containing all on-disk structures and no in-memory
{               structures.
{
```

```
{       V02-052 REFORMAT        Keith B. Thompson       29-JUL-1980
{
{       V051    jak0045         J A Krycka      21-MAR-1980     15:00
{               remove IFB$V_DAP_FMODE.
{
{       V050    cds0075         C D Saether     21-JAN-1980     11:25
{               remove IFB$L_FWA_BDB, IFB$W_BKS_BYTES, IFB$W_BKS_RECS.
{
{       V049    ras0010         R A Schaefer    07-JAN-1980     16:45
{               re-arrange drc fields for valid did check.
{
{       V048    ras0009         R A Schaefer    20-DEC-1979     15:30
{               add FWA$T_RNM_FID to hold the saved fid from a rename
{               for checking bad rename directory operation.
{
{       V047    jak0029         J A Krycka      18-NOV-1979     11:00
{               add FWA$V_REMRESULT to denote that fal has returned resultant
{               string.
{
{       v046    tmh0003         Tim Halvorsen   02-NOV-1979
{               add FWA$L_DIRBDB to hold address of directory buffer bdb
{               add FWA$V_FILEFOUND bit to indicate that at least one file
{               has been found - used to return FNF vs. NMF.
{
{       V045    jak0025         J A Krycka      28-SEP-1979     17:00
{               add FWA$V_DEV_UNDER in conjunction with effort to
{               modify expanded and resultant name string processing to prefix
{               an underscore to node and device names that have been subject
{               to logical name translation.
{
{       V044                    Tim Halvorsen   12-SEP-1979
{               add IFB$V_SEARCH bit to mark ifabs left during searches.
{
{       V043                    Tim Halvorsen   11-SEP-1979
{               overlay the fib buffer with the node descriptors to reduce
{               the total space used by fwa.  increase the size of the file
{               name and type buffers due to the new maximum sizes which can
{               occur as a result of wild characters (i.e. *a*b*c*d*e etc).
{
{       V042    jak0023         J A Krycka      27-AUG-1979     18:00
{               move FWA$T_NODEBUF to NWA$T_NODEBUF and add FWA$Q_DIR1 thru
{               FWA$Q_DIR8 in conjunction with effort in RM0XPFN to allow
{               multiple node specs in a file specification.
{
{       v041    cds0022         C Saether       26-july-79      22:45
{               add IFB$V_SEQFIL to bookkeeping bits.
{
{       V040    cds0000         C Saether       16-JAN-1979     14:00
{               remove definition of lockabove from srchflags in irab
{               and add abovelckd to bookeeping bits
{
{       V039    wsk0000         W S Koenig      10-JAN-1979     1:25
{               fix problem w/ IDX_PTR overlapping w/ AS_DEV longwords
{
{       V038    jak0017         J A Krycka      28-DEC-1978     14:30
{               make IFB$L_NWA_PTR and IFB$L_FRB_PTR separate locations so that
{               networking and file sharing do not interfere with each other.
{
{       V037    ran0003         R A Newell      20-DEC-1978     17:35
{               file sharing and isam changes.
{
{       V036    jak0007         J A Krycka      17-DEC-1978     19:00
{               change values of FWA$C_MAXNODE and FWA$C_NODBUFSIZ.
{
{       V035    jak0006         J A Krycka      21-NOV-1978     16:40
{               add FWA$V_ACCSTR definition.
{
{       V034    ran0002         R A Newell      1-NOV-1978      09:49
{               RMS32 isam modifications.  additions of isam fields, take out
{               NWA, add PLG_VER, change ASB size, define PTR_VBN, update,
{               UPD_NRP, RFA_VBN, and RFA_ID fields.
{
{++
```

```
/*
/*
/*



/*
/*
/*
/*
```

```
{
{       prologue field definitions
{
{
{  common vbn1 format for indexed and relative files
{
{  (note: indexed file defintions not included at this time)
{
module $PLGDEF;


aggregate PLGDEF structure prefix PLG$;
    FILL_1 byte dimension 11 fill prefix PLGDEF tag $$; /* leave space for indexed file things
    DBKTSIZ byte unsigned;                               /* data bucket size
    FILL_2 longword fill prefix PLGDEF tag $$;           /* filler
    FLAGS_OVERLAY union;
        FLAGS byte unsigned;                             /* flag bits
        FLAGS_BITS structure;
            NOEXTEND bitfield;                           /* no extend allowed (rel)
        end FLAGS_BITS;
    end FLAGS_OVERLAY;
    FILL_3 byte dimension 85 fill prefix PLGDEF tag $$; /* space filler
    AVBN byte unsigned;                                  /* vbn of first area descriptor
    AMAX byte unsigned;                                  /* maximum number of areas
    DVBN word unsigned;                                  /* first data bucket vbn
    FILL_4 word fill prefix PLGDEF tag $$;               /* spare
    MRN longword unsigned;                               /* maximum record number (rel)
    EOF longword unsigned;                               /* eof vbn (rel)
    VER_NO word unsigned;                                /* version number
    constant VER_NO     equals 1  prefix PLG tag $C;     /* current prolog version number
    constant VER_IDX    equals 2  prefix PLG tag $C;     /* new plg for indexed files
    constant VER_3      equals 3  prefix PLG tag $C;     /* new plg for compression, space reclamation (plg 3)
    GBC word unsigned;                                   /* default global buffer count
    constant BLN equals . prefix PLG$ tag K;
    constant BLN equals . prefix PLG$ tag C;
end PLGDEF;

end_module $PLGDEF;
```

```
module $DLCDEF;
/*
/*
/*  relative file deletion control byte bit definitions
/*


aggregate DLCDEF  union prefix DLC$;
    DLCDEF_BITS structure;
        FILL_1 bitfield length 2 fill prefix DLCDEF tag $$;/* (start with bit 2)
        DELETED bitfield mask;                          /* record deleted
        REC bitfield mask;                              /* record exists (but may have been deleted)
    end DLCDEF_BITS;
end DLCDEF;

end_module $DLCDEF;
```

```
module $BKTDEF;
/*
/* index bucket definition
/*
/* this is the bucket format for RMS-11/RMS-32 index files.
/*


aggregate BKTDEF structure prefix BKT$;
    CHECKCHAR byte unsigned;                            /* bucket check character
    AREANO_OVERLAY union;
        AREANO byte unsigned;                           /* area number form which bucket was allocated
        INDEXNO byte unsigned;                          /* index to which this bucket belongs (plg 3)
    end AREANO_OVERLAY;
    ADRSAMPLE word unsigned;                            /* address sample - low 16 bits of first vbn in bucket
    FREESPACE_OVERLAY union;
        FREESPACE word unsigned;                        /* displacement in bucket of first free byte
        KEYFRESPC word unsigned;                        /* pointer to key's free space (plg 3)
    end FREESPACE_OVERLAY;
    NXTRECID_OVERLAY union;
        NXTRECID word unsigned;                         /* next available word record id (plg 3)
        NXTRECID_FIELDS structure;
            NXTRECID byte unsigned;                     /* next available record id
            LSTRECID byte unsigned;                     /* last id in range
        end NXTRECID_FIELDS;
    end NXTRECID_OVERLAY;
    NXTBKT longword unsigned;                           /* vbn of next bucket
    LEVEL byte unsigned;                                /* bucket level number
    BKTCB_OVERLAY union;
        BKTCB byte unsigned;                            /* bucket control bits
        constant OVERHDSZ equals . prefix BKT$ tag K;   /* length of bucket overhead
        constant OVERHDSZ equals . prefix BKT$ tag C;   /* length of bucket overhead
        BKTCB_BITS structure;
            LASTBKT bitfield mask;                      /* last bucket in horizontal chain
            ROOTBKT bitfield mask;                      /* root bucket
            FILL_1 bitfield fill prefix BKTDEF tag $$;  /* spare
            PTR_SZ bitfield mask length 2;              /* size of vbn pointers in this bucket
        end BKTCB_BITS;
        constant ENDOVHD        equals 4  prefix BKT tag $C;/* end of bucket overhead
        constant DATBKTOVH      equals 2  prefix BKT tag $C;/* end of bucket overhead for data buckets
        constant DUPBKTOVH      equals 4  prefix BKT tag $C;/* additional end of data bucket overhead
                                                        /*  when duplicates are allowed (LCB pointer
        constant MAXBKTSIZ      equals 63  prefix BKT tag $C;/* maximum bucket size
    end BKTCB_OVERLAY;
end BKTDEF;

end_module $BKTDEF;
```

```
module $IRCDEF;
/*
/* index record definition
/*
/* this is the definition of RMS-11/RMS-32 index file record formats
/*



aggregate IRCDEF union prefix IRC$;
    CONTROL byte unsigned;                              /* record control byte
        CONTROL_BITS0 structure;
            PTRSZ bitfield mask length 2;               /* size of pointer
            RECORDCB bitfield mask length 6;            /* record control bits
        end CONTROL_BITS0;

/*
/* record control bits used only in primary data record and SIDR array element
/* control bytes
/*
        CONTROL_BITS1 structure;
            FILL_1 bitfield length 2 fill prefix IRCDEF tag $$;/* skip size of pointer field
            DELETED bitfield mask;                      /* record is deleted
            FILL_2 bitfield fill prefix IRCDEF tag $$;
            NOPTRSZ bitfield mask;                      /* no RRV
            FILL_3 bitfield length 2 fill prefix IRCDEF tag $$; /* skip 2 bits
            FIRST_KEY bitfield mask;
        end CONTROL_BITS1;

/*
/* record control bits used only in primary data record control bytes
/*
        CONTROL_BITS2 structure;
            FILL_3 bitfield length 3 fill prefix IRCDEF tag $$;/* skip over first 3 bits
            RRV bitfield mask;                          /* rrv record
        end CONTROL_BITS2;

/*
/* record control bits used only in prologue 2 SIDR record control bytes
/*
        CONTROL_BITS3 structure;
            FILL_5 bitfield length 4 fill prefix IRCDEF tag $$;/* skip 4 bits
            NODUPCNT bitfield mask;                     /* DUP_CNT field absent
        end CONTROL_BITS3;

/*
/* record control bits used only in prologue 3 RRV, UDR and SIDR record control
/* bytes of RU journalled files. (RU_UPDATE is set only in UDR record control
/* bytes)
/*
        CONTROL_BITS4 structure;
            FILL_6 bitfield length 5 fill prefix IRCDEF tag $$;/* skip 5 bits
            RU_DELETE bitfield mask;                    /* record is RU deleted
            RU_UPDATE bitfield mask;                    /* record is RU updated
        end CONTROL_BITS4;
```

```
/*
/* record control bits reserved for RMS-11 use only (these may not be re-defined
/* except for prologue 3 records)
/*
/* Bit number 5
/* Bit number 6
/*

/*
/* index bucket record
/*
        CONTROL_FIELDS4 structure;
            FILL_7 byte fill prefix IRCDEF tag $$;
            BUCKETPTR character length 0 tag T;           /* bucket pointer (not referenced in the code,
                                                          /*  just present for consistency)
            constant IDXPTRBAS  equals 2  prefix IRC tag $C;/* used to determine size of pointer in index
            constant IDXOVHDSZ  equals 1  prefix IRC tag $C;/* includes record control byte

/*
/* data bucket record
/*
        end CONTROL_FIELDS4;
end IRCDEF;

aggregate IRCDEF1 structure prefix IRC$;
    FILL_8 byte fill prefix IRCDEF tag $$;
    ID byte unsigned;                                     /* record id
    RRV_ID byte unsigned;                                 /* rrv's id -- always in the same place

/*
/* prologue 3 data bucket record
/*
end IRCDEF1;

aggregate IRCDEF2 structure prefix IRC$;
    FILL_9 byte fill prefix IRCDEF tag $$;
    ID word unsigned;                                     /* record id
    RRV_ID word unsigned;                                 /* rrv's id -- always in the same place
/*
/* constants
/*
    constant DATSZFLD   equals 2  prefix IRC tag $C;      /* size of size field in variable length records
    constant DATPTRBAS  equals 3  prefix IRC tag $C;      /* used to determine size of RRV in data buckets
    constant DCNTSZFLD  equals 4  prefix IRC tag $C;      /* size of duplicate count field in Plg 2 SIDRs
    constant DATOVHDSZ  equals 2  prefix IRC tag $C;      /* includes the record control byte, and the id
    constant FIXOVHDSZ  equals 7  prefix IRC tag $C;      /* the record overhead for fixed record
    constant VAROVHDSZ  equals 9  prefix IRC tag $C;      /* record overhead for variable records
    constant RRVOVHDSZ  equals 7  prefix IRC tag $C;      /* size of RRV
/*
/* prologue 3 constants
/*
    constant DATPTRBS3  equals 4  prefix IRC tag $C;      /* used to determine size of RRV in data buckets
    constant DATOVHSZ3  equals 3  prefix IRC tag $C;      /* record control byte, and id
    constant FIXOVHSZ3  equals 9  prefix IRC tag $C;      /* record overhead for fixed length records
    constant VAROVHSZ3  equals 11 prefix IRC tag $C;      /* record overhead for variable length records
    constant RRVOVHSZ3  equals 9  prefix IRC tag $C;      /* size of RRV
    constant SDROVHSZ3  equals 2  prefix IRC tag $C;     /* record overhead for SIDRs

    constant KEYCMPOVH  equals 2  prefix IRC tag $C;     /* key compression overhead
    constant DATCMPOVH  equals 3  prefix IRC tag $C;     /* data compression overhead
end IRCDEF2;

end_module $IRCDEF;
```

```
module $KEYDEF;

/*
/* definitions for the key descriptors in the prologue
/*
/* these definitions are associated w/ the plg and area definitions
/*


aggregate KEYDEF structure prefix KEY$;
    IDXFL longword unsigned;                             /* vbn for next key descriptor
    NOFF word unsigned;                                  /* offset to next key descriptor
    IANUM byte unsigned;                                 /* index area number
    LANUM byte unsigned;                                 /* level 1 area number
    DANUM byte unsigned;                                 /* data area number
    ROOTLEV byte unsigned;                               /* root level
    IDXBKTSZ byte unsigned;                              /* index bucket size
    DATBKTSZ byte unsigned;                              /* data bucket size
    ROOTVBN longword unsigned;                           /* root bucket pointer
    FLAGS_OVERLAY union;
        FLAGS byte unsigned;                             /* flag bits
        FLAGS_BITS0 structure;
            DUPKEYS bitfield mask;                       /* duplicate key values allowed
            CHGKEYS bitfield mask;                       /* key value may change on $update operation
            NULKEYS bitfield mask;                       /* null key character enabled
            IDX_COMPR bitfield mask;                     /* index is compressed
            INITIDX bitfield mask;                       /* index must be initialized
            FILL_1 bitfield fill prefix KEYDEF tag $$;   /* spare
            KEY_COMPR bitfield mask;                     /* (PLG3) key is compressed in data record
        end FLAGS_BITS0;

        FLAGS_BITS1 structure;
            FILL_2 bitfield fill prefix KEYDEF tag $$;   /* space over dupkeys
            FILL_3 bitfield length 2 fill prefix KEYDEF tag $$; /* spare
            FILL_4 bitfield fill prefix KEYDEF tag $$;   /* space over idx_compr
            FILL_5 bitfield fill prefix KEYDEF tag $$;   /* space over initidx
            FILL_6 bitfield fill prefix KEYDEF tag $$;   /* spare
            FILL_7 bitfield fill prefix KEYDEF tag $$;   /* space over key compr
            REC_COMPR bitfield mask;                     /* (PLG3) Data record is compressed
        end FLAGS_BITS1;
        constant MAX_DAT       equals 10  prefix KEY tag $C;/* (PLG3) Maximum size of a non-compressed data
                                                      /*  record
        constant MAX_PRIMARY   equals 6  prefix KEY tag $C;/* (PLG3) Maximum size of a non-compressed
                                                      /*  primary key
        constant MAX_INDEX     equals 6  prefix KEY tag $C;/* (PLG3) Maximum size of a non-compressed
                                                      /*  index and SIDR key
    end FLAGS_OVERLAY;
    DATATYPE byte unsigned;                              /* data type for key
    constant STRING    equals 0  prefix KEY tag $C;      /* string data type
    constant SGNWORD   equals 1  prefix KEY tag $C;      /* signed binary word
    constant UNSGNWORD equals 2  prefix KEY tag $C;      /* unsigned binary word
    constant SGNLONG   equals 3  prefix KEY tag $C;      /* signed binary long word
    constant UNSGNLONG equals 4  prefix KEY tag $C;      /* unsigned binary long word
    constant PACKED    equals 5  prefix KEY tag $C;      /* packed decimal
    constant SGNQUAD   equals 6  prefix KEY tag $C;      /* signed binary quadword
    constant UNSGNQUAD equals 7  prefix KEY tag $C;      /* unsigned binary quadword
    constant MAX_DATA   equals 7  prefix KEY tag $C;       /* maximum data type value allowed
    SEGMENTS byte unsigned;                                 /* number of segments in key
    NULLCHAR byte unsigned;                                 /* "null" character
    KEYSZ byte unsigned;                                    /* total key size
    KEYREF byte unsigned;                                   /* key of reference
    MINRECSZ word unsigned;                                 /* minimum record length
    IDXFILL word unsigned;                                  /* index fill quantity
    DATFILL word unsigned;                                  /* data fill quantity
    POSITION_OVERLAY union;
        POSITION word unsigned;                             /* key seg position
        POSITION0 word unsigned;                            /* another name for position 0
    end POSITION_OVERLAY;
    POSITION1 word unsigned;                                /* position 1
    POSITION2 word unsigned;                                /* position 2
    POSITION3 word unsigned;                                /* position 3
    POSITION4 word unsigned;                                /* position 4
    POSITION5 word unsigned;
    POSITION6 word unsigned;
    POSITION7 word unsigned;
    SIZE_OVERLAY union;
        SIZE byte unsigned;                                 /* key segment size
        SIZE0 byte unsigned;                                /* another name for size
    end SIZE_OVERLAY;
    SIZE1 byte unsigned;                                    /* size 1
    SIZE2 byte unsigned;
    SIZE3 byte unsigned;
    SIZE4 byte unsigned;
    SIZE5 byte unsigned;
    SIZE6 byte unsigned;
    SIZE7 byte unsigned;
    KEYNAM character length 32;                             /* key name
    LDVBN longword unsigned;                                /* first data bucket
    TYPE_OVERLAY union;
        TYPE byte unsigned;                                 /* key segment datatype (plg 3)
        TYPE0 byte unsigned;                                /* another name for first datatype (plg 3)
    end TYPE_OVERLAY;
    TYPE1 byte unsigned;                                    /* (plg 3)
    TYPE2 byte unsigned;                                    /* (plg 3)
    TYPE3 byte unsigned;                                    /* (plg 3)
    TYPE4 byte unsigned;                                    /* (plg 3)
    TYPE5 byte unsigned;                                    /* (plg 3)
    TYPE6 byte unsigned;                                    /* (plg 3)
    TYPE7 byte unsigned;                                    /* (plg 3)
    constant BLN equals . prefix KEY$ tag K;                /* length of key descriptor in the prologue (plg 3)
    constant BLN equals . prefix KEY$ tag C;                /* length of key descriptor in the prologue (plg 3)
    constant SPARE       equals 6  prefix KEY tag $C;      /* these are spare words in key block (plg 3)
end KEYDEF;

end_module $KEYDEF;
```

```
module $AREADEF;
/*
/*
/* definitions for the area descriptor in the prologue
/*
/*

aggregate AREADEF structure prefix AREA$;
    FILL_1 byte fill prefix AREADEF tag $$;            /* spare
    FLAGS byte unsigned;                               /* not currently used
    AREAID byte unsigned;                              /* area id
    ARBKTSZ byte unsigned;                             /* bucket size for area
    VOLUME word unsigned;                              /* relative volume number
    ALN byte unsigned;                                 /* extend allocation alignment
    constant CYL        equals 1  prefix AREA tag $C;  /* cylinded alignment
    constant LBN        equals 2  prefix AREA tag $C;  /* logical block alignment
    constant VBN        equals 3  prefix AREA tag $C;  /* virtual block alignment
    constant RFI        equals 4  prefix AREA tag $C;  /* allocate close to related file by fid
    AOP_OVERLAY union;
        AOP byte unsigned;                             /* alignment options
        AOP_BITS structure;
            HARD bitfield mask;                        /* absolute alignment or nothing
            ONC bitfield mask;                         /* locate on cylinder
            FILL_2 bitfield length 3 fill prefix AREADEF tag $$;
            CBT bitfield mask;                         /* contiguous best try
            FILL_3 bitfield fill prefix AREADEF tag $$;
            CTG bitfield mask;                         /* contiguous
        end AOP_BITS;
    end AOP_OVERLAY;
    AVAIL longword unsigned;                           /* available (returned) buckets
    CVBN longword unsigned;                            /* start vbn for current extent
    CNBLK longword unsigned;                           /* number of blocks in current extent
    USED longword unsigned;                            /* number of blocks used
    NXTVBN longword unsigned;                          /* next vbn to use
    NXT longword unsigned;                             /* start vbn for next extent
    NXBLK longword unsigned;                           /* number of blocks in next extent
    DEQ word unsigned;                                 /* default extend quantity
    FILL_4 byte dimension 2 fill prefix AREADEF tag $$; /* spare
    LOC longword unsigned;                             /* start lbn on volume
    RFI word unsigned dimension 3;                     /* related file id
    TOTAL_ALLOC longword unsigned;                     /* total block allocation
    FILL_5 byte dimension 8 fill prefix AREADEF tag $$;/* spare
    CHECK word unsigned;                               /* checksum
    constant BLN equals . prefix AREA$ tag K;          /* length of area descriptor in the prologue
    constant BLN equals . prefix AREA$ tag C;          /* length of area descriptor in the prologue
end AREADEF;

end_module $AREADEF;
```

```
module $RJRDEF;
/*
/*
/* definitions for the journaling records in RMS journals
/*      for performance reasons, the BKT and BLK forms
/*      should be an integral number of quadwords.
/*
/*

aggregate RJRDEF structure prefix RJR$;
    FLAGS_OVERLAY union;                                    /* control flags
        FLAGS word unsigned;
        FLAGS_BITS structure;
        FILL_1 bitfield length 16 fill prefix RJRDEF tag $$;
        end FLAGS_BITS;
    end FLAGS_OVERLAY;
    VERSION byte unsigned;                                  /* RMS journal version #
        constant VER1   equals 1  prefix RJR tag $C;        /* journal version 1
        constant VER2   equals 2  prefix RJR tag $C;        /* journal version 2
        constant MAXVER equals 2  prefix RJR tag $C;        /* version limit
    ENTRY_TYPE byte unsigned;                               /* journal entry type
        constant NOENT          equals 0 prefix RJR tag $C; /* null type
        constant MAPPING        equals 1 prefix RJR tag $C; /* mapping entry
        constant FILENAME       equals 1 prefix RJR tag $C; /* mapping entry synonym
        constant RECORD         equals 2 prefix RJR tag $C; /* record entry
        constant BLOCK          equals 3 prefix RJR tag $C; /* block IO entry (at, etc...)
        constant BUCKET         equals 4 prefix RJR tag $C; /* ISAM bucket
        constant EXTEND         equals 5 prefix RJR tag $C; /* extend (AT, AI)
        constant AT_RECORD      equals 6 prefix RJR tag $C; /* audit trail record
        constant MAXTYP         equals 6 prefix RJR tag $C; /* entry-type limit
    ORG byte unsigned;                                      /* file organization
        constant SEQ    equals 0  prefix RJR tag $C;        /* sequential file org
        constant REL    equals 1  prefix RJR tag $C;        /* relative file org
        constant IDX    equals 2  prefix RJR tag $C;        /* indexed file org
        constant HSH    equals 3  prefix RJR tag $C;        /* hashed file org
        constant MAXORG equals 3  prefix RJR tag $C;        /* org limit
    OPER byte unsigned;                                     /* RMS operation id
        constant NOOP           equals 0   prefix RJR tag $;    /* null operation
        constant BUCKET         equals 1   prefix RJR tag $;    /* bucket-level I/O
        constant CLOSE          equals 2   prefix RJR tag $;    /* close
        constant CONNECT        equals 3   prefix RJR tag $;    /* connect
        constant CREATE         equals 4   prefix RJR tag $;    /* create
        constant DELETE         equals 5   prefix RJR tag $;    /* delete
        constant DISCONNECT     equals 6   prefix RJR tag $;    /* disconnect
        constant DISPLAY        equals 7   prefix RJR tag $;    /* display
        constant ENTER          equals 8   prefix RJR tag $;    /* enter
        constant ERASE          equals 9   prefix RJR tag $;    /* erase
        constant EXTEND         equals 10  prefix RJR tag $;    /* extend
        constant FIND           equals 11  prefix RJR tag $;    /* find
        constant FLUSH          equals 12  prefix RJR tag $;    /* flush
        constant FREE           equals 13  prefix RJR tag $;    /* free
        constant GET            equals 14  prefix RJR tag $;    /* get
        constant MODIFY         equals 15  prefix RJR tag $;    /* modify
        constant NXTVOL         equals 16  prefix RJR tag $;    /* next volume
        constant OPEN           equals 17  prefix RJR tag $;    /* open
        constant PARSE          equals 18  prefix RJR tag $;    /* parse
        constant PUT            equals 19  prefix RJR tag $;     /* put
        constant READ           equals 20  prefix RJR tag $;     /* block I/O read
        constant RELEASE        equals 21  prefix RJR tag $;     /* release
        constant REMOVE         equals 22  prefix RJR tag $;     /* remove
        constant RENAME         equals 23  prefix RJR tag $;     /* rename
        constant REWIND         equals 24  prefix RJR tag $;     /* rewind
        constant SEARCH         equals 25  prefix RJR tag $;     /* search
        constant SPACE          equals 26  prefix RJR tag $;     /* block I/O space
        constant TRUNCATE       equals 27  prefix RJR tag $;     /* truncate
        constant UPDATE         equals 28  prefix RJR tag $;     /* update
        constant WAIT           equals 29  prefix RJR tag $;     /* wait
        constant WRITE          equals 30  prefix RJR tag $;     /* block I/O write
        constant TPT            equals 31  prefix RJR tag $;     /* truncate on PUT
        constant MAXOPER        equals 31  prefix RJR tag $;     /* oper limit
    JNL_TYPE byte unsigned;                                      /* journaling type
        constant NOJNL  equals 0 prefix RJR tag $C;              /* null jnl type
        constant RMS_AI equals 1 prefix RJR tag $C;              /* after-image journal
        constant RMS_BI equals 2 prefix RJR tag $C;              /* before-image journal
        constant RMS_RU equals 3 prefix RJR tag $C;              /* recovery unit
        constant MAXJNL equals 3 prefix RJR tag $C;              /* jnl type limit
    FILL_C2 byte dimension 1 fill prefix RJRDEF tag $$;          /* spare
    JNLID_OVERLAY union fill;                                    /* RMS journal ID
        JNLID character length 28;                               /*
    JNLID_FIELDS structure fill;
        VOLNAM character length 12;                              /* volume name
        FID character length 6;                                  /* file id
        FILL_C3 word fill prefix RJRDEF tag $$;
        ID_DATE quadword unsigned;                               /* time
    end JNLID_FLAGS;
    end JNLID_OVERLAY;
    AT_STS longword unsigned;                                    /* status of operation
    AT_STV longword unsigned;                                    /* secondary status
    AT_CTX longword unsigned;                                    /* user FAB/RAB CTX field
    FILL_C4 longword fill prefix RJRDEF tag $$;
    FILL_C5 longword fill prefix RJRDEF tag $$;
    constant HDRLEN equals . prefix RJR$ tag K;                  /* common header len
    constant HDRLEN equals . prefix RJR$ tag C;                  /* common header len

    ENTRY_OVERLAY union;

/*
/* The FILENAME entry is used to describe filename information required to:
/*
/* 1.   Identify a stream of journal entries with a particular file.
/*      Used in particular for AI volume recovery or roll back RUs on remount.
/*
/* 2.   Record the information required to re-create a file for AI journaling.
/*
/* 3.   Record all the required audit-trail information for a $CREATE.
/*

        FILENAME_ENTRY structure;
            FILL_F1 longword fill prefix RJRDEF tag $$;
            ATR_FLAGS_OVERLAY union;
                ATR_FLAGS longword unsigned;
                ATR_FLAGS_BITS structure;
```

```
                    ATR_UCHAR bitfield;                 /* UCHAR attribute present
                    ATR_PROT bitfield;                  /* PROT attribute present
                    ATR_UIC bitfield;                   /* UIC attribute present
                    ATR_REC bitfield;                   /* RECORD attributes present
                    ATR_EXPIRE bitfield;                /* EXPIRATION present
                end ATR_FLAGS_BITS;
            end ATR_FLAGS_OVERLAY;
            UIC longword unsigned;                      /* owner UIC
            PROT longword unsigned;                     /* prot mask
            ALLOC longword unsigned;                    /* initial allocation (audit)
            UCHAR longword unsigned;                    /* user characteristics (create)
            EXPIRE quadword unsigned;                   /* expiration date (create)
            FNS byte unsigned;                          /* size of file name
            FILL_F2 byte dimension 1 fill prefix RJRDEF tag $$;
            FAC byte unsigned;                          /* file access (audit)
            SHR byte unsigned;                          /* sharing allowed (audit)
            DID word unsigned dimension 3;              /* directory ID (create, volume recovery)
            FILL_F3 word fill prefix RJRDEF tag $$;
            C_FIB character length 64;                  /* FIB (create)
            constant C_FIBLEN equals 64 prefix RJR$ tag K;
            constant C_FIBLEN equals 64 prefix RJR$ tag C;
            REC_ATTR character length 32;               /* record attributes (create)
            constant RECATRLEN equals 32 prefix RJR$ tag K;
            constant RECATRLEN equals 32 prefix RJR$ tag C;
            FILENAME character length 256;              /* full filename
            constant FILNAMLEN equals . prefix RJR$ tag K;
            constant FILNAMLEN equals . prefix RJR$ tag C;
        end FILENAME_ENTRY;

        RECORD_ENTRY structure;                         /* record entry
            FILL_REC1 longword fill prefix RJRDEF tag $$;
            CHKSUM longword unsigned;                   /* checksum of old record
            RFA_OVERLAY union;
                RFA word unsigned dimension 3;          /* RFA of record
                RFA_FIELDS structure;
                    RFA0 longword unsigned;             /* alternate RFA def
                    RFA4 word unsigned;
                end RFA_FIELDS;
                RRN longword unsigned;                  /* relative record number
            end RFA_OVERLAY;
            RSIZE word unsigned;                        /* record size
            RIMAGE character length 0;                  /* record date
            constant RECLEN equals . prefix RJR$ tag K; /* record entry len
            constant RECLEN equals . prefix RJR$ tag C; /* record entry len
        end RECORD_ENTRY;

/*
/* The block entry is common to both AT and block IO journaling.
/*

        BLOCK_ENTRY structure;                          /* block entry
            FILL_BLK1 longword fill prefix RJRDEF tag $$;
            BLOCK_VBN longword unsigned;                /* vbn of block
            BLOCK_SIZE longword unsigned;               /* transfer size (AT)
            BLOCK character length 0;                   /* block data
            constant BLKLEN equals . prefix RJR$ tag K; /* block i/o entry len
        constant BLKLEN equals . prefix RJR$ tag C; /* block i/o entry len
    end BLOCK_ENTRY;

/*
/* The extend entry is common to both AT and AI journaling.
/*

    EXTEND_ENTRY structure;                         /* extend entry
        FILL_EXT1 longword fill prefix RJRDEF tag $$;
        FILL_EXT2 character length 32;              /* no longer FIB
                                                    /* currently unused
        EXT_FLAGS_OVERLAY union;
            EXT_FLAGS longword unsigned;
            EXT_FLAGS_BITS structure;
                EXT_USE_XAB bitfield;               /* ALL XAB fields present
            end EXT_FLAGS_BITS;
        end EXT_FLAGS_OVERLAY;
        FILL_EXT3 longword fill prefix RJRDEF tag $$;
/*
/* Fields EXT_AOP (unused) through EXT_RFI are in same relative locations as
/* the same fields in allocation XAB.
/*
        EXT_AOP byte unsigned;              /* align options
        EXT_ALN byte unsigned;              /* alignment boundary
        EXT_VOL word unsigned;              /* relative volume number
        EXT_LOC longword unsigned;          /* location
        EXT_ALQ longword unsigned;          /* allocation quantity
        EXT_DEQ word unsigned;              /* default extension
        EXT_BKZ byte unsigned;              /* bucket size
        EXT_AID byte unsigned;              /* area ID
        EXT_RFI word unsigned dimension 3;  /* related file IFI
        EXT_ENDALL character length 0;      /* end of all info

        constant EXTLEN equals . prefix RJR$ tag K; /* extend entry len
        constant EXTLEN equals . prefix RJR$ tag C; /* extend entry len
    end EXTEND_ENTRY;

    BUCKET_ENTRY structure;                         /* BUCKET entry
        FILL_BKT1 longword fill prefix RJRDEF tag $$;
        BKT_VBN longword unsigned;                  /* bucket vbn
        BKT_SIZE word unsigned;                     /* bucket size
        JBKT_SIZE word unsigned;                    /* actual size of
                                                    /* journaled bucket data
        BUCKET character length 0;                  /* bucket data
        constant BKTLEN equals . prefix RJR$ tag K; /* bucket entry len
        constant BKTLEN equals . prefix RJR$ tag C; /* bucket entry len
    end BUCKET_ENTRY;

    AT_RECORD structure;
        FILL_AT1 longword fill prefix RJRDEF tag $$;
        AT_ROP longword unsigned;                   /* record options
        AT_KRF byte unsigned;                       /* key of reference
        AT_KSZ byte unsigned;                       /* key size
        AT_RAC byte unsigned;                       /* record access mode
        FILL_AT2 byte fill prefix RJRDEF tag $$;
        AT_RFA_OVERLAY union;
```

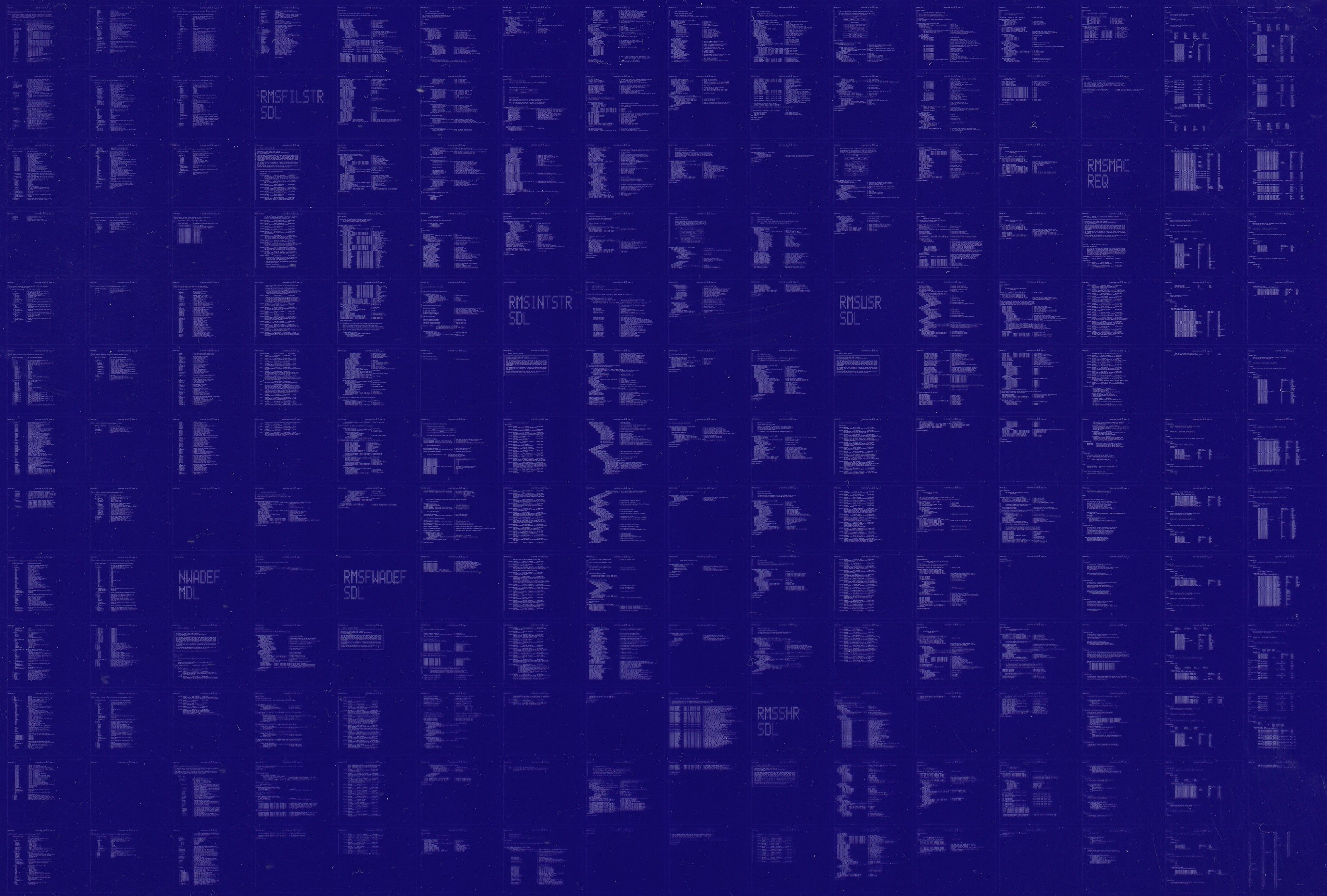
```
                AT_RFA word unsigned dimension 3;       /* RFA of record
                AT_RFA_FIELDS structure;
                    AT_RFA0 longword unsigned;          /* alternate RFA def
                    AT_RFA4 word unsigned;
                end AT_RFA_FIELDS;
                AT_RRN longword unsigned;               /* relative record number
            end AT_RFA_OVERLAY;
            FILL_AT3 word fill prefix RJRDEF tag $$;
            AT_KEY character length 0;                  /* key if used
            constant AT_RECLEN equals . prefix RJR$ tag K;
            constant AT_RECLEN equals . prefix RJR$ tag C;
        end AT_RECORD;

    end ENTRY_OVERLAY;
    constant BLN equals . prefix RJR$ tag K;            /* length of RJR descriptor in the prologue
    constant BLN equals . prefix RJR$ tag C;            /* length of RJR descriptor in the prologue
end RJRDEF;

end_module $RJRDEF;
```

0313 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



RMSFILSTR
SDL

RMSMAC
REQ

RMSINTSTR
SDL

RMSUSR
SDL

NWADEF
MDL

RMSFWADEF
SDL

RMSSHR
SDL